A Method for Image Reduction Based on a Generalization of Ordered Weighted Averaging Functions

Journal:	*Transactions on Fuzzy Systems*
Manuscript ID	TFS-2015-0927
Manuscript Type:	Full Paper
Date Submitted by the Author:	15-Dec-2015
Complete List of Authors:	Farias, Antonio Diego; Universidade Federal Rural do Semi-Arido, Mathematics Costa, Valdigleis S.; Federal University of Rio Grande do Norte, Department of Informatics and Applied Mathematics Lopes, Luiz Ranyer; Federal University of Rio Grande do Norte, Department of Informatics and Applied Mathematics Bedregal, Benjamin; Federal University of Rio Grande do Norte, Department of Informatics and Applied Mathematics Santiago, Regivan; Federal University of Rio Grande do Norte, Department of Informatics and Applied Mathematics
Keywords:	Aggregation functions, OWAs, Dynamic OWAs, Image Reduction

A Method for Image Reduction Based on a Generalization of Ordered Weighted Averaging Functions

A. Diego S. Farias*†, Valdigleis S. Costa†, Luiz Ranyer A. Lopes†, Benjamín Bedregal† and Regivan H. N. Santiago†

Abstract

In this paper we propose a special type of aggregation function which generalizes the notion of Ordered Weighted Averaging Function - *OWA*. The resulting functions are called **Dynamic Ordered Weighted Averaging Functions — DYOWA**s. This generalization will be developed in such way that the weight vectors are variables depending on the input vector. Particularly, this operators generalize the aggregation functions: *Minimum*, *Maximum*, *Arithmetic Mean*, *Median* etc, which are extensively used in image processing. In this field of research two problems are considered: The determination of methods to reduce images and the construction of techniques which provide noise reduction. The operators described here are able to be used in both cases. In terms of image reduction we apply the methodology provided in [1]. We use the noise reduction operators obtained here to treat the images obtained in the first part of the paper, thus obtaining images with better quality.

Index Terms

Aggregation functions, *OWA* functions, Image reduction, Noise reduction

I. Introduction

Image processing has great applicability in several areas. In medicine, for example, they can be applied to: Identify tumors [2]; support techniques in advancing dental treatments [3], etc. Such images are not always obtained with a suitable quality, and to detect the desired information, various methods have been developed in order to eliminate most of the noise contained in these images.

Another problem addressed in image processing is the decrease of resolution, usually aiming the reduction of memory consumption required for its storage [4].

*Federal University of Semi-Arid - UFERSA, Pau dos Ferros - RN, 59.900-000, †Department of Informatics and Applied Mathematics, Federal University of Rio Grande do Norte (UFRN), Natal – RN, 59.072-970, E-mail addresses: antonio.diego@ufersa.edu.br, valdigleis@ppgsc.ufrn.br, ranyer.lopes@gmail.com, bedregal@dimap.ufrn.br, regivan@dimap.ufrn.br

There are several techniques for image reduction in the literature, more recently Paternain *et. al.* [1] constructed reduction operators using weighted averaging aggregation functions. The proposed method consists of: (1) To reduce a given image by using a reduction operator; (2) To build a new image from the reduced one, and (3) To analyze the quality of the last image by using the measures *PSNR* and *MSIM* [4].

In this work we introduce a class of aggregation functions called: **Dynamic Ordered Weighted Averaging Function - (DYOWA)**. They generalize the OWA function introduced by Yager [5] and in particular the operators: *Arithmetic Mean*, *Median*, *Maximum*, *Minimum* and $cOWA$. We provide a range of their properties such as: idempotence, symmetry and homogeneity as well two methods [1]: **(1)** for image reduction and **(2)** for noise treatment.

This paper is structured in the following way: SECTION 2 provides some basics of Aggregation Functions Theory. SECTION 3 introduces Dynamic Ordered Weighted Averaging functions, shows some examples and properties, and introduces a particular $DYOWA$ function, called **H**, which will be fundamental for this work. In SECTION 4 we provide an application of $DYOWA$'s to image reduction and in SECTION 5, we show that $DYOWA$ functions are able to treat images with noise, aiming to improve the reduction method used in SECTION 4. Finally, section SECTION 6 gives the final remarks of this work.

II. Aggregation Functions

Aggregation functions are important mathematical tools for applications in several fields: Information fuzzy [6]; Decision making [7]–[11]; Image processing [1], [2], [12] and Engineering [13]. In this section we introduce them together with examples and properties. We also present a special family of aggregation functions called *Ordered Weighted Averaging* - **OWA** and show some of its features.

A. Definition and Examples

Aggregation functions associate each entry $\mathbf{x}$ with n arguments in the closed interval $[0,1]$ an output value also in the interval $[0,1]$; formally we have:

Definition 1. *An n-ary aggregation function is a mapping $f : [0,1]^n \rightarrow [0,1]$, which associates each n-dimensional vector $\mathbf{x}$ to a single value $f(\mathbf{x})$ in the interval $[0,1]$ such that:*

1) $f(0,...,0) = 0$ and $f(1,...,1) = 1$;

[1]These methods were implemented by using Java 1.8.0_31 software in a 64 bits MS Windows machine.

2) If $\mathbf{x} \leq \mathbf{y}$, i.e., $x_i \leq y_i$, for all $i = 1, 2, ..., n$, then $f(\mathbf{x}) \leq f(\mathbf{y})$.

Example 1.

(a) *Arithmetic Mean:* $Arith(\mathbf{x}) = \frac{1}{n}(x_1 + x_2 ... + x_n)$

(b) *Minimum:* $Min(\mathbf{x}) = min\{x_1, x_2, ..., x_n\}$*;*

(c) *Maximum:* $Max(\mathbf{x}) = max\{x_1, x_2, ..., x_n\}$*;*

(d) *Harmonic mean:* $f_n(\mathbf{x}) = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}}$*;*

From now on we will use the short term "aggregation" instead of "n-ary aggregation function".

Aggregations can be divided into four distinct classes: *Averaging, Conjunctive, Disjunctive* and *Mixed*. Since this paper focus on averaging aggregations, we will define only this class. A wider approach in aggregation can be found in [14]–[17].

Definition 2. *An aggregation is called* ***Averaging****, if for all* $\mathbf{x} \in [0, 1]^n$ *we have:*

$$Min(\mathbf{x}) \leq f(\mathbf{x}) \leq Max(\mathbf{x})$$

Example 2. *The Arithmetic Mean, the Maximum and the Minimum are averaging aggregation functions.*

B. Special Types Aggregation Functions

An aggregation function f:

(1) is **Idempotent** if, and only if, $f(x, ..., x) = x$ for all $x \in [0, 1]$.

(2) is **Homogeneous** of order k if, and only if, for all $\lambda \in [0, 1]$ and $\mathbf{x} \in [0, 1]^n$, $f(\lambda x_1, \lambda x_2, ..., \lambda x_n) = \lambda^k f(x_1, x_2, ..., x_n)$. When f is homogeneous of order 1 we simply say that f is homogeneous.

(3) is **Shift-invariant** if, and only if, $f(x_1 + r, x_2 + r, .., x_n + r) = f(x_1, x_2, .., x_n) + r$, for all $r \in [-1, 1]$, $\mathbf{x} \in [0, 1]^n$ such that $(x_1 + r, x_2 + r, ..., x_n + r) \in [0, 1]^n$ and $f(x_1, x_2, ..., x_n) + r \in [0, 1]$.

(4) is **Monotonic** if, and only if, $\mathbf{x} \leq \mathbf{y}$ implies $f(\mathbf{x}) \leq f(\mathbf{y})$.

(5) is **Strictly Monotone** if, and only if, $f(\mathbf{x}) < f(\mathbf{y})$ whenever $\mathbf{x} < \mathbf{y}$, i.e. $\mathbf{x} \leq \mathbf{y}$ and $\mathbf{x} \neq \mathbf{y}$.

(6) has a **Neutral Element** $e \in [0, 1]$, if for all $t \in [0, 1]$ at any coordinate input vector $\mathbf{x}$, it has to be:

$$f(e, ..., e, t, e, ..., e) = t, \text{ and}$$

(7) f is **Symmetric** if, and only if, its value is not changed under the permutations of the coordinates

of **x**, i.e, we have:

$$f(x_1, x_2, ..., x_n) = f(x_{p_{(1)}}, x_{p_{(2)}}, \cdots, x_{p_{(n)}})$$

For all x and any permutation $P : \{1, 2, ..., n\} \to \{1, 2, ..., n\}$.

(8) An **Absorbing Element (*Annihilator*)** of an aggregation function f, is an element $a \in [0, 1]$ such that:

$$f(x_1, x_2, ..., x_{i-1}, a, x_{i+1}, ..., x_n) = a$$

(9) A **Zero Divisor** of an aggregation function is an element $a \in]0, 1[$, such that, there is some vector **x** with $x_j > 0$, for all $1 \le j \le n$, and $f(x_1, ..., x_{j-1}, a, x_{j+1}, .., x_n) = 0$.

(10) A **One Divisor** of an aggregation function f is an element $a \in]0, 1[$ such that, there is some vector **x** with $x_j < 1$, for all $1 \le j \le n$, and $f(x_1, ..., x_{j-1}, a, x_{j+1}, .., x_n) = 1$.

(11) If $N : [0, 1] \to [0, 1]$ is a strong negation[2] and $f : [0, 1]^n \to [0, 1]$ is an aggregation function, then the **dual aggregation function** of f is:

$$f^d(x_1, x_2, ..., x_n) = N(f(N(x_1), N(x_2), ..., N(x_n)))$$

which is also an aggregation function.

Example 3.

(i) *The functions: $Arith, Min$ and Max are examples of idempotent, homogeneous, shift-invariant, monotonic and symmetric functions.*

(ii) *Min and Max have 0 and 1 elements as annihilator, respectively, but $Arith$ does not have annihiladors.*

(iii) *Min, Max and $Arith$ do not have zero divisors and one divisors.*

(iv) *The dual of Max with respect to negation $N(x) = 1 - x$ is the Min function.*

C. Ordered Weighted Averaging Function - OWA

In the field of aggregation functions there is a very important subclass in which the elements are parametric; they are called: ***Ordered Weighted Averaging*** or simply **OWA** [5].

An OWA is an aggregation function which associates weights to all components x_i of an input vector **x**. To achieve that observe the following definition.

[2]A **strong negation** is an antitonic function $N : [0, 1] \to [0, 1]$ such that $N(N(\alpha)) = \alpha$ for all $\alpha \in [0, 1]$.

Definition 3. *Let be an input vector* $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in [0,1]^n$ *and a vector of weights* $\mathbf{w} = (w_1, \ldots, w_n)$*, such that* $\sum_{i=1}^{n} w_i = 1$*. Assuming the permutation:*

$$Sort(\mathbf{x}) = (x_{p(1)}, x_{p(2)}, \ldots, x_{p(n)})$$

such that $x_{p(i)} \geq x_{p(i+1)}$*, i.e.,* $x_{p(1)} \geq x_{p(2)} \geq \ldots \geq x_{p(n)}$*, the Ordered Weighted Averaging (OWA) Function with respect to* $\mathbf{w}$*, is the function* $OWA_{\mathbf{w}} : [0,1]^n \to [0,1]$ *such that:*

$$OWA_{\mathbf{w}}(\mathbf{x}) = \sum_{i=1}^{n} w_i \cdot x_{p(i)}$$

In what follows we remove $\mathbf{w}$ from $OWA_{\mathbf{w}}(\mathbf{x})$. The main properties of such functions are:

(a) For any vector of weights $\mathbf{w}$, the function $OWA(\mathbf{x})$ is idempotent and monotonic. Moreover, $OWA(\mathbf{x})$ is strictly increasing if all weights $\mathbf{w}$ are positive;

(b) The dual of a $OWA_{\mathbf{w}}$ is denoted by $(OWA)^d$, with the vector of weights dually ordered, i.e. $(OWA_{\mathbf{w}})^d = OWA_{\mathbf{w}^d}$, where $\mathbf{w}^d = (w_{p(n)}, w_{p(n-1)}, ..., w_{p(1)})$.

(c) OWA are continuous, symmetric and shift-invariant functions;

(d) They do not have neutral or absorption elements, except in the special case of functions OWA of Max and Min.

1) Examples of special functions OWA:

1. If all weight vector components are equal to $\frac{1}{n}$, then $OWA(\mathbf{x}) = Arith((\mathbf{x})$.
2. If $\mathbf{w} = (1, 0, 0, ..., 0)$, then $OWA(\mathbf{x}) = Max(\mathbf{x})$.
3. If $\mathbf{w} = (0, 0, 0, ..., 1)$, then $OWA(\mathbf{x}) = Min(\mathbf{x})$.
4. if $w_i = 0$, for all i, with the exception of a k-th member, i.e, $w_k = 1$, then this OWA is called **static** and $OWA_{\mathbf{w}}(x) = x_k$
5. Given a vector $\mathbf{x}$ and its ordered permutation $Sort(\mathbf{x}) = (x_{(1)}, \ldots, x_{(n)})$, the *Median* function

$$Med(\mathbf{x}) = \begin{cases} \frac{1}{2}(x_{(k)} + x_{(k+1)}), & \text{if } n = 2k \\ x_{(k+1)}, & \text{if } n = 2k+1 \end{cases}$$

is an OWA function in which the vector of weights is defined by:

- If n is odd, then $w_i = 0$ for all $i \neq \lceil \frac{n}{2} \rceil$ and $w_{\lceil n/2 \rceil} = 1$.
- If n is even, then $w_i = 0$ for all $i \neq \lceil \frac{n+1}{2} \rceil$ and $i \neq \lfloor \frac{n+1}{2} \rfloor$, and $w_{\lceil n/2 \rceil} = w_{\lfloor n/2 \rfloor} = \frac{1}{2}$.

Example 4. *The* n*-dimensional cOWA function [18] is the OWA operator, with weighted vector defined* *by:*

- *If n is even, then* $w_j = \frac{2(2j-1)}{n^2}$, *for* $1 \leq j \leq \frac{n}{2}$, *and* $w_{n/2+i} = w_{n/2-i+1}$, *for* $1 \leq i \leq \frac{n}{2}$.
- *If n is odd, then* $w_j = \frac{2(2j-1)}{n^2}$, *for* $1 \leq j \leq \frac{n-1}{2}$, $w_{n/2+i} = w_{n/2-i+1}$, *for* $1 \leq i \leq \frac{n}{2}$, *and* $w_{(n+1)/2} = 1 - 2\sum\limits_{j=1}^{(n-1)/2} w_i$.

The OWA functions are defined in terms of a predetermined vector of weights. In the next section we propose the generalization of the concept of OWA in order to relax the vector of weights. To achieve that we replace the vector of weights by a family of functions. The resulting functions are called **Dynamic Ordered Weighted Avegaring Functions** or in short: **DYOWA**s.

III. Dynamic Ordered Weighted Avegaring Functions - $DYOWA$

Before defining the notion of $DYOWA$ functions, we need to establish the notion of *weight-function*.

Definition 4. *A finite family of functions* $\Gamma = \{f_i : [0,1]^n \to [0,1] \mid 1 \leq i \leq n\}$ *such that:*

$$\sum_{i=1}^{n} f_i(\mathbf{x}) = 1.$$

is called family of **weight-function** *(FWF).*

A **Dynamic Ordered Weighted Averaging** *Function or simply* **DYOWA** *associated to a FWF* Γ *is a function of the form:*

$$DYOWA_\Gamma(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x}) \cdot x_i$$

Below we show some examples of $DYOWA$ operators with their respective weight-functions.

Example 5. *Let be* $\Gamma = \{f_i(\mathbf{x}) = \frac{1}{n} \mid 1 \leq i \leq n\}$. *The $DYOWA$ operator associated to* Γ, $DYOWA_\Gamma(\mathbf{x})$, *is* $Arith(\mathbf{x})$.

Example 6. *The function Minimum can be obtained from* $\Gamma = \{f_i \mid 1 \leq i \leq n\}$, *where* $f_1(\mathbf{x}) = f_2(\mathbf{x}) = \cdots = f_{n-1}(\mathbf{x}) = 0$ *and* $f_n(\mathbf{x}) = 1$, *for all* $\mathbf{x} \in [0,1]^n$.

Example 7. *Similarly, the function Maximum is also of type $DYOWA$ with Γ dually defined.*

Example 8. *For any vector of weights* $\mathbf{w} = (w_1, w_2, ..., w_n)$, *A function* $OWA_\mathbf{w}(\mathbf{x})$ *is a $DYOWA$ in which the weight-functions are given by:* $f_i(\mathbf{x}) = w_{p(i)}$, *where* $p : \{1, 2, \cdots, n\} \longrightarrow \{1, 2, \cdots, n\}$ *is the permutation, such that* $p(i) = j$ *with* $x_i = x_{(j)}$. *For example: If* $\mathbf{w} = (0.3, 0.4, 0.3)$, *then for* $\mathbf{x} =$

$(0.1, 1.0, 0.9)$ *we have* $x_1 = x_{(3)},\ x_2 = x_{(1)}$ *and* $x_3 = x_{(2)}$. *Thus,* $f_1(\mathbf{x}) = 0.3,\ f_2(\mathbf{x}) = 0.3,\ f_3(\mathbf{x}) = 0.4$, *and* $DYOWA(\mathbf{x}) = 0.3 \cdot 0.1 + 0.3 \cdot 1.0 + 0.4 \cdot 0.9 = 0.69$

Remark 1. *Example 8 shows that the functions OWA, introduced by Yager, are special cases of* $DYOWA$ *functions. There are, however, some* $DYOWA$ *functions which are not OWA.*

Example 9. *Let* $\Gamma = \{\sin(x) \cdot y, 1 - \sin(x) \cdot y\}$. *The respective* $DYOWA$ *function is* $DYOWA(x, y) = (\sin(x) \cdot y) \cdot x + (1 - \sin(x) \cdot y) \cdot y$, *which is not an OWA function.*

A. Properties of $DYOWA$ *Functions*

The next theorem characterizes the $DYOWA$ functions which are also aggregations.

Theorem 1. *Let* $\Gamma = \{f_1, \cdots, f_n\}$ *be a FWF. A* $DYOWA_\Gamma$ *is an aggregation function if, and only if, it is monotonic.*

Proof. Obviously, if $DYOWA_\Gamma$ is an aggregation, then it is monotonic function. Conversely, if $DYOWA_\Gamma$ is monotonic, then for it to become an aggregation, enough to show that

$$DYOWA_\Gamma(0, ..., 0) = 0 \text{ e } DYOWA_\Gamma(1, ..., 1) = 1,$$

this follows from the definition of $DYOWA$. □

Corollary 1. *A* $DYOWA$ *is an aggregation function if, and only if, it is an a aggregation of type averaging.*

Proof. For all $\mathbf{x} = (x_1, ..., x_n)$ have to

$$Min(\mathbf{x}) \leq x_i \leq Max(\mathbf{x}),\ \forall i = 1, 2, ..., n.$$

So,

$$\sum_{i=1}^{n} f_i(\mathbf{x}) \cdot Min(\mathbf{x}) \leq \sum_{i=1}^{n} f_i(\mathbf{x}) \cdot x_i \leq \sum_{i=1}^{n} f_i(\mathbf{x}) \cdot Max(\mathbf{x}),$$

but as $\sum_{i=1}^{n} f_i(\mathbf{x}) = 1$, it follows that

$$Min(\mathbf{x}) \leq \sum_{i=1}^{n} f_i(\mathbf{x}) \cdot x_i \leq Max(\mathbf{x})$$

□

Corollary 2. *All functions of the type $DYOWA$ presented in examples 4, 5, 6, 7 and 8 are averaging aggregation functions.*

Proof. Just see that those functions are monotonic. □

Proposition 1. *For every Γ, $DYOWA_\Gamma$ is idempotent.*

Proof. If $\mathbf{x} = (x, ..., x)$ with $t \in [0, 1]$, then:

$$DYOWA_\Gamma(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x}) \cdot x = x \cdot \sum_{i=1}^{n} f_i(\mathbf{x}) = x$$

□

This property is important because it tells us that every $DYOWA$ is idempotent, regardless it is an aggregation or not.

Proposition 2. *If Γ is invariant under translations, i.e, $f_i(x_1 + \lambda, x_2 + \lambda, ..., x_n + \lambda) = f_i(x_1, x_2, ..., x_n)$ for any $\mathbf{x} \in [0, 1]^n$, for $i \in \{1, 2, \cdots, n\}$ and $\lambda \in [-1, 1]$, then $DYOWA_\Gamma$ is shift-invariant.*

Proof. Let $\mathbf{x} = (x_1, ..., x_n) \in [0, 1]^n$ and $\lambda \in [-1, 1]$ such that $(x_1 + \lambda, x_2 + \lambda, ..., x_n + \lambda) \in [0, 1]^n$. then,

$$\begin{aligned} DYOWA_\Gamma(x_1 + \lambda, ..., x_n + \lambda) &= \\ = \quad & \sum_{i=1}^{n} f_i(x_1 + \lambda, ..., x_n + \lambda) \cdot (x_i + \lambda) \\ = \quad & \sum_{i=1}^{n} f_i(x_1 + \lambda, ..., x_n + \lambda) \cdot x_i \\ & + \sum_{i=1}^{n} f_i(x_1 + \lambda, ..., x_n + \lambda) \cdot \lambda \\ = \quad & \sum_{i=1}^{n} f_i(x_1, ..., x_n) \cdot x_i + \lambda \\ = \quad & DYOWA_\Gamma(x_1, ..., x_n) + \lambda \end{aligned}$$

□

Proposition 3. *If Γ is homogeneous of order k (i.e., if f_i is homogeneus of order k, for each $f_i \in \Gamma$), then $DYOWA_\Gamma(\mathbf{x})$ is homogeneous of order $k + 1$.*

Proof. Of course that, if $\lambda = 0$, then $DYOWA_\Gamma(\lambda x_1, ..., \lambda x_n) = \lambda f(x_1, ..., x_n)$. Now, to $\lambda \neq 0$ we have:

$$\begin{aligned} DYOWA_\Gamma(\lambda x_1, ..., \lambda x_n) &= \sum_{i=1}^{n} f_i(\lambda x_1, ..., \lambda x_n) \cdot \lambda x_i \\ &= \lambda \cdot \sum_{i=1}^{n} \lambda^k f_i(x_1, ..., x_n) x_i \\ &= \lambda^{k+1} \cdot DYOWA_\Gamma(x_1, ..., x_n) \end{aligned}$$

□

Example 10. *Let* Γ *be defined by*

$$f_i(x_1, ..., x_n) = \begin{cases} \frac{1}{n}, & \textit{if } x_1 = ... = x_n = 0 \\ \frac{x_i}{\sum_{j=1}^{n} x_j}, & \textit{otherwise} \end{cases}$$

Then,

$$DYOWA_\Gamma(\mathbf{x}) = \begin{cases} 0, & \textit{if } x_1, ..., x_n = 0 \\ \frac{\sum_{i=1}^{n} x_i^2}{\sum_{i=1}^{n} x_i}, & \textit{otherwise} \end{cases}$$

This $DYOWA_\Gamma$ *is idempotent, homogeneous and shift-invariant. However,* $DYOWA_\Gamma$ *is not monotonic, since* $DYOWA_\Gamma(0.5, 0.2, 0.1) = 0.375$ *and* $DYOWA_\Gamma(0.5, 0.22, 0.2) = 0.368$.

The next definition provides a special FWF, which will be used to build a $DYOWA$ whose properties are very important for this paper.

Definition 5. *Consider the family* Γ *of functions*

$$f_i(\mathbf{x}) = \begin{cases} \frac{1}{n}, & \textit{if } \mathbf{x} = (x, ..., x) \\ \frac{1}{n-1}\left(1 - \frac{|x_i - Med(\mathbf{x})|}{\sum_{j=1}^{n} |x_j - Med(\mathbf{x})|}\right), & \textit{otherwise} \end{cases}$$

Then, Γ *is a FWF, i.e.* $\sum_{i=1}^{n} f_i(\mathbf{x}) = 1$*, for all* $\mathbf{x} \in [0,1]^n$*. Let* $\mathbf{H}$ *be the associated* $DYOWA$*. The computation of* $\mathbf{H}$ *can be performed using the following expressions:*

$$\mathbf{H}(\mathbf{x}) = \begin{cases} x, & \textit{if } \mathbf{x} = (x, ..., x) \\ \frac{1}{n-1} \sum_{i=1}^{n} \left(x_i - \frac{x_i |x_i - Med(\mathbf{x})|}{\sum_{j=1}^{n} |x_j - Med(\mathbf{x})|} \right), & \textit{otherwise} \end{cases}$$

Example 11. *Let be $n = 3$. So, for $\mathbf{x} = (0.1, 0.3, 0)$ we have*

$$f_1(\mathbf{x}) = 0.5,\ \ f_2(\mathbf{x}) = 0.167,\ \ f_3(\mathbf{x}) = 0.333 \text{ and } \mathbf{H}(\mathbf{x}) = 0.08.$$

Proposition 4. *The weight-functions defined in Definition 5 are such that: $f_i(x_1 + \lambda, ..., x_n + \lambda) = f_i(x_1, x_2, ..., x_n)$ and $f_i(\lambda x_1, ..., \lambda x_n) = f_i(x_1, ..., x_n)$, for any $1 \leq i \leq n$.*

Proof. Writing $\mathbf{x}' = (x_1+\lambda, ..., x_n+\lambda)$, then $f(x_1+\lambda, ..., x_n+\lambda) = (f_1(\mathbf{x}'), ..., f_n(\mathbf{x}'))$. Clearly, $Med(\mathbf{x}') = Med(\mathbf{x}) + \lambda$. Thus, for $\mathbf{x} \neq (x, ..., x)$ we have:

$$\begin{aligned} f_i(\mathbf{x}') &= \frac{1}{n-1}\left(1 - \frac{|x_i+\lambda-Med(\mathbf{x}')|}{\sum\limits_{j=1}^{n}|x_j+\lambda-Med(\mathbf{x}')|}\right) \\ &= \frac{1}{n-1}\left(1 - \frac{|x_i+\lambda-(Med(\mathbf{x})+\lambda)|}{\sum\limits_{j=1}^{n}|x_j+\lambda-(Med(\mathbf{x})+\lambda)|}\right) \\ &= \frac{1}{n-1}\left(1 - \frac{|x_i-Med(\mathbf{x})|}{\sum\limits_{j=1}^{n}|x_j-Med(\mathbf{x})|}\right) \\ &= f_i(\mathbf{x}). \end{aligned}$$

Therefore, $f(\mathbf{x}') = (f_1(\mathbf{x}'), ..., f_n(\mathbf{x}')) = (f_1(\mathbf{x}), ..., f_n(\mathbf{x}))$. The case in which $\mathbf{x} = (x, ..., x)$ is immediate.

To check the second property, make $\mathbf{x}'' = (\lambda x_1, ..., \lambda x_n)$, note that $Med(\mathbf{x}'') = \lambda med(\mathbf{x})$ and for $\mathbf{x} \neq (x, ..., x)$

$$\begin{aligned} f_i(\mathbf{x}'') &= \frac{1}{n-1}\left(1 - \frac{|\lambda x_i-Med(\lambda\mathbf{x})|}{\sum\limits_{j=1}^{n}|\lambda x_j-Med(\lambda\mathbf{x})|}\right) \\ &= \frac{1}{n-1}\left(1 - \frac{|\lambda x_i-\lambda Med(\mathbf{x})|}{\sum\limits_{j=1}^{n}|\lambda x_j-\lambda Med(\mathbf{x})|}\right) \\ &= \frac{1}{n-1}\left(1 - \frac{|\lambda|\cdot|x_i-Med(\mathbf{x})|}{|\lambda|\cdot\sum\limits_{j=1}^{n}|x_j-Med(\mathbf{x})|}\right) \\ &= \frac{1}{n-1}\left(1 - \frac{|x_i-Med(\mathbf{x})|}{\sum\limits_{j=1}^{n}|x_j-Med(\mathbf{x})|}\right) \\ &= f_i(\mathbf{x}) \end{aligned}$$

Therefore, $f(\mathbf{x}'') = (f_1(\mathbf{x}''), ..., f_n(\mathbf{x}'')) = (f_1(\mathbf{x}), ..., f_n(\mathbf{x})) = f(\mathbf{x})$. The case in which $\mathbf{x} = (x, ..., x)$ is also immediately □

Corollary 3. $\mathbf{H}$ *is shift-invariant and homogeneous.*

Proof. Straightforward for propositions 2 and 3. □

The function $\mathbf{H}$ is of great importance to this work, since this function, as well as some $DYOWA$'s already mentioned will provide us tools able: (1) To reduce the size of images and (2) To deal with noise reduction.

Now, we present some other properties of function $\mathbf{H}$.

B. Properties of $\mathbf{H}$

In addition to idempotency, homogeneity and shift-invariance $\mathbf{H}$ has the following proprerties.

Proposition 5. $\mathbf{H}$ *has no neutral element.*

Proof. Suppose $\mathbf{H}$ has a neutral element e, find the vector of weight for $\mathbf{x} = (e, ..., e, x, e, ..., e)$. Note that if $n \geq 3$, then $Med(\mathbf{x}) = e$ and therefore,

$$\begin{aligned} f_i(\mathbf{x}) &= \tfrac{1}{n-1}\left(1 - \tfrac{|x_i - Med(\mathbf{x})|}{\sum\limits_{j=1}^{n} |x_j - Med(\mathbf{x})|}\right) \\ &= \tfrac{1}{n-1}\left(1 - \tfrac{|x_i - e|}{\sum\limits_{j=1}^{n} |x_j - e|}\right) \\ &= \tfrac{1}{n-1}\left(1 - \tfrac{|x_i - e|}{|x - e|}\right) \end{aligned}$$

therefore,

$$f_i(\mathbf{x}) = \begin{cases} \frac{1}{n-1}, \text{ if } x_i = e \\ 0, \text{ if } x_i = x \end{cases}, \text{ to } n \geq 3$$

i.e.,

$$f(\mathbf{x}) = \left(\tfrac{1}{n-1}, ..., \tfrac{1}{n-1}, 0, \tfrac{1}{n-1}, ..., \tfrac{1}{n-1}\right)$$

and

$$\mathbf{H}(\mathbf{x}) = (n-1) \cdot \frac{e}{n-1} = e$$

But since e is a neutral element of $\mathbf{H}$, $\mathbf{H}(\mathbf{x}) = x$. Absurd, since we can always take $x \neq e$.

For $n = 2$, we have $Med(\mathbf{x}) = \dfrac{x+e}{2}$, where $\mathbf{x} = (x, e)$ or $\mathbf{x} = (e, x)$. In both cases it is not difficult to show that $f(\mathbf{x}) = (0.5, 0.5)$ and $\mathbf{H}(\mathbf{x}) = \dfrac{x+e}{2}$. Thus, taking $x \neq e$, again we have $\mathbf{H}(x, e) \neq x$. □

Proposition 6. **H** *has no absorbing elements.*

Proof. To $n = 2$, we have $\mathbf{H}(\mathbf{x}) = \frac{x_1 + x_2}{2}$, which has no absorbing elements. Now for $n \geq 3$ we have to $\mathbf{x} = (a, 0, ..., 0)$ with $Med(\mathbf{x}) = 0$ therefore,

$$f_1(\mathbf{x}) = \frac{1}{n-1}\left(1 - \frac{a}{a}\right) = 0 \text{ and } f_i = \frac{1}{n-1}, \forall i = 2, ..., n.$$

therefore,

$$\mathbf{H}(a, 0, ..., 0) = 0 \cdot a + \frac{1}{n-1} \cdot 0 + ... + \frac{1}{n-1} \cdot 0 = a \Rightarrow a = 0,$$

but to $\mathbf{x} = (a, 1, ..., 1)$ we have to $Med(\mathbf{x}) = 1$. Furthermore,

$$f_1(\mathbf{x}) = \frac{1}{n-1}\left(1 - \frac{1-a}{1} - a\right) = 0$$

and

$$f_i = \frac{1}{n-1} \text{ para } i = 2, 3, ..., n.$$

therefore,

$$\mathbf{H}(a, 1, ..., 1) = 0 \cdot a + \frac{1}{n-1} \cdot 1 + ... + \frac{1}{n-1} \cdot 1 = a \Rightarrow a = 1.$$

With this we prove that **H** does note have annihiladors. □

Proposition 7. **H** *has no zero divisors.*

Proof. Let $a \in]0, 1[$ and consider $\mathbf{x} = (a, x_2, ..., x_n) \in]0, 1]^n$. In order to have $\mathbf{H}(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x}) \cdot x_i = 0$ we have $f_i(\mathbf{x}) \cdot x_i = 0$ for all $i = 1, 2, ..., n$. But as $a \neq 0$ and we can always take $x_2, x_3, ..., x_n$ also different from zero, then for each $i = 1, 2, ..., n$ there remains only the possibility of terms:

$$f_i(\mathbf{x}) = 0 \text{ para } i = 1, 2, ..., n.$$

This is absurd, for $f_i(\mathbf{x}) \in [0, 1]$ e $\sum_{i=1}^{n} f_i(\mathbf{x}) = 1$. like this, **H** has no zero divisors. □

Proposition 8. **H** *does not have one divisors*

Proof. Just to see that $a \in]0, 1[$, we have to $\mathbf{H}(a, 0, ..., 0) = f_1(\mathbf{x}).a \leq a < 1$. □

Proposition 9. **H** *is symmetric.*

Proof. Let $P : \{1, 2, ..., n\} \rightarrow \{1, 2, ..., n\}$ be a permutation. So we can easily see that $Med(x_{P(1)}, x_{P(2)}, ..., x_{P(n)}) =$

$Med(x_1, x_2, ..., x_n)$ for all $\mathbf{x} = (x_1, x_2, ..., x_n) \in [0,1]^n$. We also have to $\sum_{i=1}^{n} |x_{P(i)} - Med(x_{P(1)}, x_{P(2)}, ..., x_{P(n)})| = \sum_{i=1}^{n} |x_i - Med(\mathbf{x})|$. Thus, it suffices to consider the case where $(x_{P(1)}, x_{P(2)}, ..., x_{P(n)}) \neq (x, x, ..., x)$. But $(x_{P(1)}, x_{P(2)}, ..., x_{P(n)}) \neq (x, x, ..., x)$ we have to:

$$\begin{aligned}
&\mathbf{H}(x_{P(1)}, x_{P(2)}, ..., x_{P(n)}) = \\
&= \frac{1}{n-1} \sum_{i=1}^{n} \left(x_{P(i)} - \frac{x_{P(i)} |x_{P(i)} - Med(x_{P(1)}, ..., x_{P(n)})|}{\sum_{j=1}^{n} |x_{P(i)} - Med(x_{P(1)}, ..., x_{P(n)})|} \right) \\
&= \frac{\sum_{i=1}^{n} x_{P(i)}}{n-1} - \frac{1}{n-1} \cdot \sum_{i=1}^{n} \frac{x_{P(i)} |x_{P(i)} - Med(x_1, ..., x_n)|}{\sum_{j=1}^{n} |x_{P(i)} - Med(x_1, ..., x_n)|} \\
&= \frac{\sum_{i=1}^{n} x_i}{n-1} - \frac{1}{n-1} \cdot \sum_{i=1}^{n} \frac{x_{P(i)} |x_{P(i)} - Med(x_1, ..., x_n)|}{\sum_{j=1}^{n} |x_i - Med(x_1, ..., x_n)|} \\
&= \frac{\sum_{i=1}^{n} x_i}{n-1} - \frac{1}{n-1} \cdot \sum_{i=1}^{n} \frac{x_i |x_i - Med(x_1, ..., x_n)|}{\sum_{j=1}^{n} |x_i - Med(x_1, ..., x_n)|} \\
&= \mathbf{H}(x_1, ..., x_n).
\end{aligned}$$

□

Therefore, $\mathbf{H}$ satisfies the following properties:

- Idempotence
- Homogeneity
- Shift-invariance
- Symmetry.
- $\mathbf{H}$ has no neutral element
- $\mathbf{H}$ has no absorbing elements
- $\mathbf{H}$ has no zero divisors
- $\mathbf{H}$ does not have one divisors

Remark 2. *Unfortunately we do not prove here the monotonicity of* $\mathbf{H}$*, due to its complexity, but we suspect that it is true. This demonstration will be relegated to a future work.*

The next two sections show the suitability of $DYOWA$. They will provide applications for image and noise reduction.

IV. $DYOWA$'S AS IMAGES REDUCTION TOOLS

In this part of our work we use the functions $DYOWA$ studied in Examples 4, 5, 6, 7 and 8, and definition 5 to build image reduction operators, the resulting images will be compared with the reduced image obtained from the operator function $\mathbf{H}$.

An image is a matrix $m \times n$, $M = A(i,j)$, where each $A(i,j)$ represents a pixel. In essence, a reduction operator reduces a given image $m \times n$ to another $m' \times n'$, such that $m' < m$ and $n' < n$. For example,

$$\begin{bmatrix} 0.1 & 0.2 & 0 & 0.5 \\ 0.3 & 0.3 & 0.2 & 0.8 \\ 1 & 0.5 & 0.6 & 0.4 \\ 0 & 0.3 & 0.5 & 0.7 \end{bmatrix} \longmapsto \begin{bmatrix} 0.1 & 0 \\ 1 & 0.6 \end{bmatrix}$$

In Grayscale images the value of pixels belong to the set $[0, 255]$, which can be normalized by dividing them by 255, so that we can think of pixels as values in the range $[0, 1]$.

Fig. 1. Example of image in Grayscale.

There are several possible ways to reduce a given image, as shown in the following example:

Example 12. *The image*

$$M = \begin{bmatrix} 0.8 & 0.7 & 0.2 & 1 & 0.5 & 0.5 \\ 0.6 & 0.2 & 0.3 & 0.1 & 1 & 0 \\ 0 & 0 & 0.6 & 0.4 & 0.9 & 1 \\ 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 \end{bmatrix},$$

can be reduced to another 2×3 *by partitioning* M *in blocks* 2×2*:*

$$\overline{M} = \begin{bmatrix} \begin{bmatrix} 0.8 & 0.7 \\ 0.6 & 0.2 \end{bmatrix} & \begin{bmatrix} 0.2 & 1 \\ 0.3 & 0.1 \end{bmatrix} & \begin{bmatrix} 0.5 & 0.5 \\ 1 & 0 \end{bmatrix} \\ \\ \begin{bmatrix} 0 & 0 \\ 0.1 & 0.2 \end{bmatrix} & \begin{bmatrix} 0.6 & 0.4 \\ 0.3 & 0.4 \end{bmatrix} & \begin{bmatrix} 0.9 & 1 \\ 0.5 & 0.6 \end{bmatrix} \end{bmatrix},$$

and applying to each block, for example, the function $f(x, y, z, w) = Max(x, y, z, w)$*: We obtain, the image:*

$$M_* = \begin{bmatrix} 0.8 & 1 & 1 \\ 0.2 & 0.6 & 1 \end{bmatrix}$$

Applying $g(x, y, z, w) = Min(x, y, z, w)$ *we would obtain:*

$$M_{**} = \begin{bmatrix} 0.2 & 0.1 & 0 \\ 0 & 0.2 & 0.5 \end{bmatrix}$$

In fact, if we apply any other function, we get a new image (usually different from the previous one), but what is the best?

One possible answer to this question involves a method called **magnification** or **extension** (see [19]–[21]), which is a method which magnifies the reduced image to another with the same size of the original one. The magnified image is then compared with the original input image.

Example 13. *From* M_* *and* M_{**}*, we get images* 4×6*,* M' *and* M''*, simply cloning each pixel ,*

$$\begin{bmatrix} x \end{bmatrix} \longmapsto \begin{bmatrix} x & x \\ x & x \end{bmatrix}$$

We obtain, new images

$$M_1 = \begin{bmatrix} 0.8 & 0.8 & 1 & 1 & 1 & 1 \\ 0.8 & 0.8 & 1 & 1 & 1 & 1 \\ 0.2 & 0.2 & 0.6 & 0.6 & 1 & 1 \\ 0.2 & 0.2 & 0.6 & 0.6 & 1 & 1 \end{bmatrix}$$

and

$$M_2 = \begin{bmatrix} 0.2 & 0.2 & 0.1 & 0.1 & 0 & 0 \\ 0.2 & 0.2 & 0.1 & 0.1 & 0 & 0 \\ 0 & 0 & 0.2 & 0.2 & 0.5 & 0.5 \\ 0 & 0 & 0.2 & 0.2 & 0.5 & 0.5 \end{bmatrix}$$

Since M_1 e M_2 have the same size as the original image M, we can now measure what is the best reduction. This can be done by comparing the initial image M with each of the resulting images, M_1 and M_2. But, how do we compare?

One of the possibilities to compare the images M_1 and M_2 with the original image M is to use the mensure PSNR [4], calculated as follows:

$$PSNR(I,K) = 10 \cdot log_{10}\left(\frac{MAX_I^2}{MSE(I,K)}\right),$$

where $I = I(i,j)$ and $K = K(i,j)$ are two images, $MSE(I,K) = \frac{1}{nm}\sum_{i=1}^{m}\sum_{j=1}^{n}[I(i,j) - K(i,j)]^n$ and MAX_I is the maximum possible pixel value of pixel. Observe that the closer the image the smaller the value of MSE and the larger the value of PSNR [3].

In what follows, we use $DYOWA$ operators: $\mathbf{H}, cOWA, Median$ and $Arith$ to reduce size of images in grayscale. We apply the following method:

Method 1

1) Reduce the input images using the **H**, *cOWA*, *Arithmetic Mean* and *Median*;
2) Magnify the reduced image to the size of the original image using the method described in example 13;
3) Compare the last image with the original one using the measure $PSNR$.

Remark 3. *This general method can be applied to any kind of image. In this work we applied it to the 10 images in grayscale of size* 512×512 *(Figure 2)* [4].

In the tables I and II we present the PSNR values between the input images and the output provided by Method 1. Table 1 provides results for operators using blocks 2×2 and Table II for blocks 4×4.

According to PSNR, $Arith$ provided the higher quality image. However, the reduction operators generated by $\mathbf{H}$ and $cOWA$ provide us quite similar images to those given by $Arith$.

[3]In particular, if the input image are equal, then the MSE value is zero and the PSNR will be infinity.
[4]In this paper we made two reductions: using 2×2 blocks and 4×4 blocks.

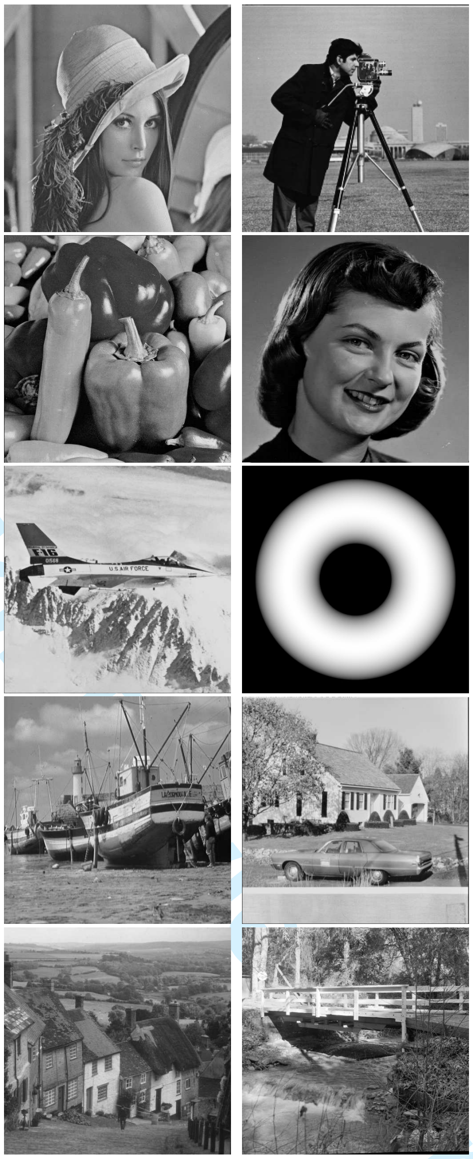

Fig. 2. Original images

	H	*cOWA*	*Arith*	*Median*
Img 01	29.63	29.66	**29, 71**	29.50
Img 02	33.15	33.14	**33.18**	33.09
Img 03	29.52	29.53	**29.57**	29.44
Img 04	31.54	31.54	**31.61**	31.46
Img 05	27.87	27.88	**27.91**	27.80
Img 06	40.78	40.78	**40.79**	40.78
Img 07	27.40	27.42	**27.47**	27.30
Img 08	26.56	26.57	**26.61**	26.47
Img 09	28.84	28.85	**28.89**	28.73
Img 10	24.43	24.45	**24.53**	24.27
Avg	29.97	29.98	**30.03**	29.88

TABLE I
$PSNR$ VALUES AFTER A REDUCTION USING THE $DYOWA$S OPERATORS USING BLOCKS 2×2

	H	*cOWA*	*Arith*	*Median*
Img 01	26.34	26.26	26.36	**26.70**
Img 02	23.64	23.60	**23.65**	22.78
Img 03	25.55	25.46	**25.56**	24.84
Img 04	27.53	27.45	**27.54**	26.86
Img 05	**24.14**	24.06	**24.14**	23.28
Img 06	34.39	34.34	**34.41**	33.83
Img 07	23.98	23.88	**23.99**	23.19
Img 08	**23.07**	22.97	**23.07**	22.18
Img 09	25.78	25.69	**25.79**	25.05
Img 10	**21.71**	21.61	**21.71**	20.62
Avg	25.61	25.53	**25.62**	24.83

TABLE II
$PSNR$ VALUES AFTER A REDUCTION USING THE $DYOWA$S OPERATORS USING BLOCKS 4×4.

Observe that although the Method 1 is very simple, it introduces noise in the resulting image. In what follows we show that the operator $\mathbf{H}$ is suitable to filter images with noise. This is done by using $\mathbf{H}$ to define the weights which are used in the process of convolution. This new process will, then, be used to provide a better comparison in the Method 1.

V. $DYOWA$'s as Tools of Noise Reduction

In this section we show that the $DYOWA$ operators studied in section III can be used to deal with images containing noise.

The methodology employed here consists to analyze the previous images with Gaussian noise $\sigma = 10\%$ and 15%; apply a filter built upon the operators $\mathbf{H}$, $cOWA$ and $Arith$ based on convolution method (See [4]), and compare the resulting images with the original one using PSNR.

	H	*cOWA*	*Arith*	No Tratament
Img 01	**30.96**	30.56	**30.96**	23.83
Img 02	**28.16**	27.78	27.36	24.36
Img 03	**31.33**	30.99	31.08	24.23
Img 04	**32.33**	32.09	32.20	24.48
Img 05	**30.39**	30.09	30.10	24.06
Img 06	31.66	**31.69**	31.38	25.73
Img 07	**28.97**	28.65	28.80	23.93
Img 08	**28.51**	28.28	28.25	24.02
Img 09	**30.03**	29.73	30.02	23.91
Img 10	**25.97**	25.84	25.81	23.76
Avg	**29.83**	29.57	29.60	24.23

TABLE III
$PSNR$ VALUES BETWEEN THE OUTPUT IMAGE WITH ORIGINAL ONE, IN WHICH $\sigma = 10\%$

	H	*cOWA*	*Arith*	No Tratament
Img 01	29.88	29.40	**29.97**	21.19
Img 02	**27.33**	27.04	26.74	21.48
Img 03	**29.92**	29.55	29.79	21.32
Img 04	**30.23**	30.06	30.08	21.51
Img 05	**29.38**	28.96	29.27	21.30
Img 06	27.95	**28.03**	27.56	22.36
Img 07	**28.23**	27.84	28.14	21.26
Img 08	**27.87**	27.57	27.70	21.28
Img 09	29.17	28.76	**29.22**	21.25
Img 10	**25.55**	25.39	25.44	21.41
Avg	**28.55**	28.26	28.39	21.44

TABLE IV
$PSNR$ VALUES BETWEEN THE OUTPUT IMAGE WITH ORIGINAL ONE, IN WHICH $\sigma = 15\%$.

Tables III and IV demonstrate the power of $\mathbf{H}$ on images with noise. All listed operators improved significantly the quality of the image with noise. However, $\mathbf{H}$ exceeded all other analyzed.

Figure 4 shows an example of a image with Gaussian noise $\sigma = 15\%$ and the Figure 5 the output image after applying the filter of convolution using $\mathbf{H}$.

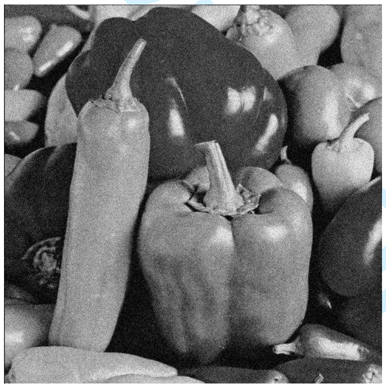

Fig. 3. Image 03 with Gaussian noise $\sigma = 15\%$.

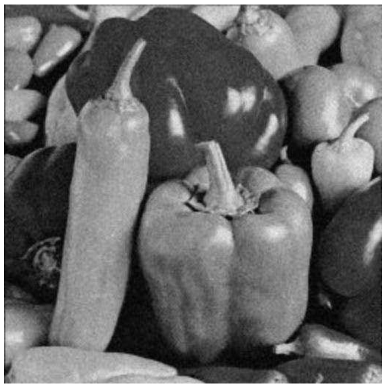

Fig. 4. Figure 4 after being treated with H by convolution.

The reader can see in tables III and IV that $\mathbf{H}$ proved to be an excellent operator for noise reduction.

In what follows, we modify the Method 1 in order to provide a better magnified image to be compare with the original one.

Method 1'

1) Reduce the input images using the $\mathbf{H}$, $cOWA$, *Arithmetic Mean* and *Median*;
2) (A) Magnify the reduced image to the size of the original image using the method described in Example 13, and (B) Use the convolution filter, using $\mathbf{H}$, on the last image;
3) Compare the last image with the original one using the measure $PSNR$.

Tables V and VI show the obtained results:

	H	*cOWA*	*Arith*	*Median*
Img 01	30.60	30.55	**30.62**	30.50
Img 02	28.68	28.67	**28.70**	28.68
Img 03	**30.89**	30.83	30.85	30.84
Img 04	**32.74**	32.71	32.71	32.71
Img 05	**29.22**	29.18	29.16	29.19
Img 06	43.88	43.80	**43.93**	43.90
Img 07	**28.05**	28.01	28.03	27.95
Img 08	**27.39**	27.37	27.37	27.36
Img 09	29.36	29.35	**29.40**	29.30
Img 10	24.94	24.92	**24.95**	24.83
Avg	**30.57**	30.54	**30.57**	30.53

TABLE V

$PSNR$ BETWEEN THE ORIGINAL IMAGE AND THE MAGNIFIED IMAGE FROM THE IMAGE REDUCED BY BLOCKS 2×2.

	H	*cOWA*	*Arith*	*Median*
Img 01	**27.44**	27.41	27.41	27.01
Img 02	**23.91**	23.88	**23.91**	23.24
Img 03	**26.86**	26.85	26.85	26.28
Img 04	**28.87**	28.86	28.85	28.39
Img 05	**25.15**	**25.15**	25.12	24.64
Img 06	**28.13**	28.05	**28.13**	27.04
Img 07	**24.68**	24.63	**24.68**	24.13
Img 08	**23.78**	23.76	**23.78**	23.14
Img 09	**26.45**	26.40	**26.45**	25.92
Img 10	**22.27**	22.21	22.26	21.49
Avg	**25.75**	25.72	25.74	25.12

TABLE VI

$PSNR$ BETWEEN THE ORIGINAL IMAGE AND THE MAGNIFIED IMAGE FROM THE IMAGE REDUCED BY BLOCKS 4×4

Since the output of convolution using $\mathbf{H}$ is closer to the original input image, the tables V and VI show that the process of reduction using $\mathbf{H}$ is more efficient.

VI. Final Remarks

In this paper we propose a generalized form of Ordered Weighted Averaging function, called **Dynamic Ordered Weighted Averaging** function or simply **DYOWA**. This functions are defined by weights, which are obtained dynamically from of each input vector $\mathbf{x} \in [0,1]^n$. We demonstrate, among other results, that OWA functions are instances of $DYOWA$s, and, hence, functions like: *Arithmetic Mean*, *Median*, *Maximum*, *Minimum* and $cOWA$ are also examples of $DYOWA$.

In the second part of this work we present a particular $DYOWA$, called of **H**, and show that it is idempotent, symmetric, homogeneous, shift-invariant, and moreover, it has no zero divisors and one divisors, and also does not have neutral elements. Since aggregation functions which satisfy these properties are extensively used in image processing, we tested its usefulness to: (1) reduce the size of images and (2) deal with noise in images.

In terms of image reduction, Method 1 showed a weakness, since it adds noise during the process of magnification. However, the treatment of noise with function **H** improved the magnification step providing an evidence that the function **H** is more efficient to perform the image reduction process.

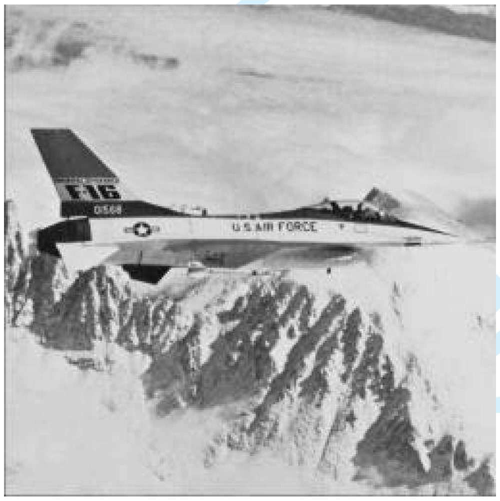

Fig. 5. Magnification of image 06 reduce by bloks 2×2 using the operator H by Method 1

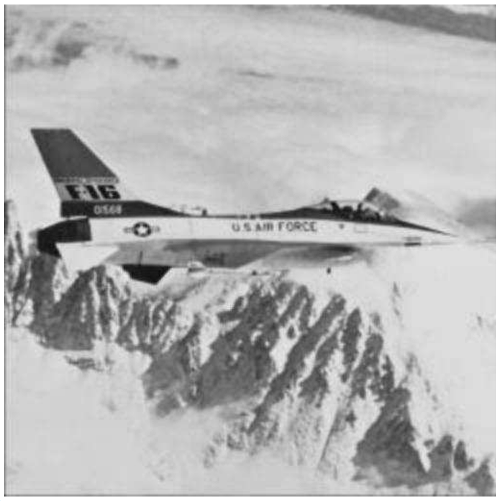

Fig. 6. Magnification of image 06 reduce by bloks 2×2 using the operator H by Method 1'

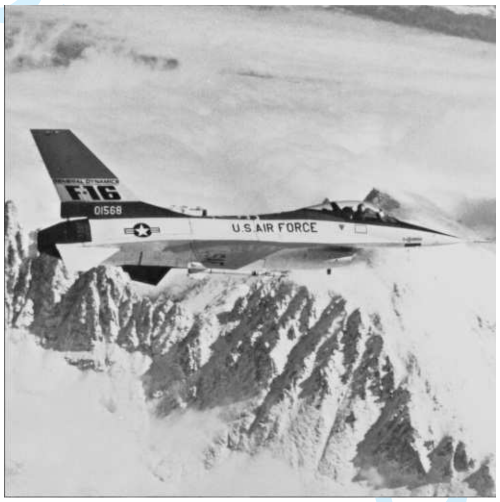

Fig. 7. Image 06

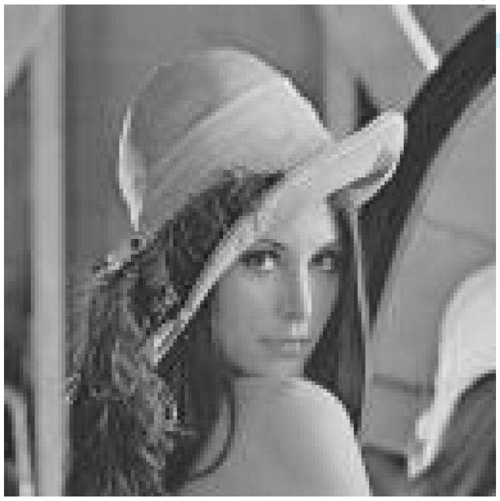

Fig. 8. Magnification of image 01 reduce by bloks 4×4 using the operator H by Method 1

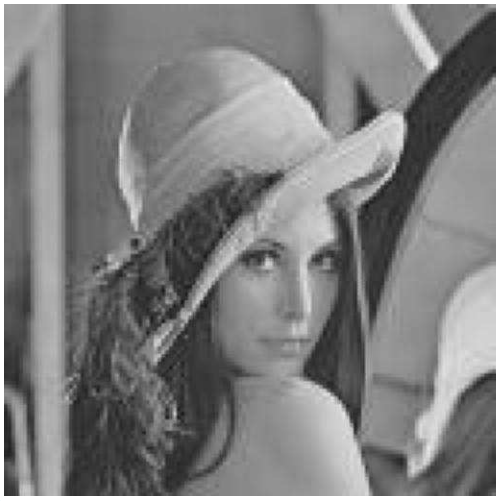

Fig. 9. Magnification of image 01 reduce by bloks 4×4 using the operator H by Method 1'

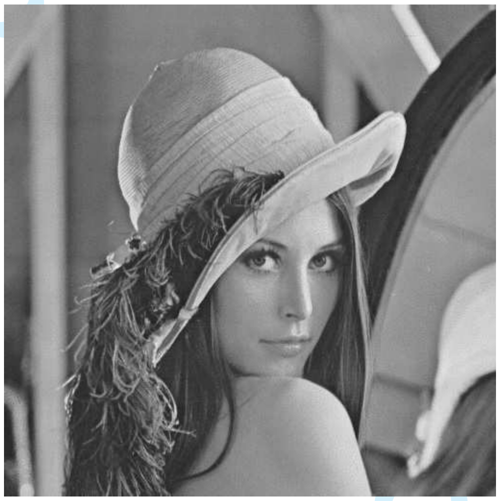

Fig. 10. Image 01

References

[1] D. Paternain, J. Fernandeza, H. Bustince, R. Mesiar ,G. Beliakov, Construction of image reduction operators using averaging aggregation functions, Fuzzy Sets and Systems 261 (2015) 87–111.

[2] R. P. Joseph, C. S. Singh, M. Manikandan, Brain Tumor MRI Image Segmentation and Detection in Image Processing, International Journal of Research and Tecnology, vol 3 (2014), ISSN: 2319-1163.

[3] A. J. Solanki, K. R. Jain, N. P. Desai, ISEF Based Identification of RCT/Filling in Dental Caries of Decayed Tooth, International Journal of Image Processing (IJIP), vol 7 (2013) 149-162.

[4] R. C. Gonzales, R. E. Woods, Digital Image Processing, third edition, Pearson, 2008.

[5] R. R. Yager, Ordered weighted averaging aggregation operators in multicriteria decision making, IEEE Trans. Syst. ManCybern. 18 (1988) 183-190.

[6] D. Dubois, H. Prade, On the use of aggregation operations in information fusion processes, Fuzzy Sets Syst. 142 (2004) 143-161.

[7] S.-J. Chen and C.-L. Hwang. Fuzzy Multiple Attribute Decision Making: Methods and Applications. Springer, Berlin, Heidelberg, 1992.

[8] S. -M. Zhou, F. Chiclana, R. I. John, J. M. Garibaldi, Type - 1 OWA operators for aggregating uncertain information with uncertain weight sinduced by type -2 linguistic quantifiers, Fuzzy Sets Syst. 159 (2008) 3281-3296.

[9] R. R. Yager, G. Gumrah, M. Reformat, Using a web personal evaluation tool — PET for lexicographic multi-criteria service selection, Knowl. - Based Syst. 24 (2011) 929-942.

[10] D. Paternain, A. Jurio, E. Barrenechea, H. Bustince, B. C. Bedregal, E. Szmidt: An alternative to fuzzy methods in decision-making problems. Expert Syst. Appl. 39(9): 7729-7735 (2012).

[11] H. Bustince, M. Galar, B. C. Bedregal, A. Kolesárová, R. Mesiar: A New Approach to Interval-Valued Choquet Integrals and the Problem of Ordering in Interval-Valued Fuzzy Set Applications. IEEE T. Fuzzy Systems 21(6): 1150-1162 (2013).

[12] G. Beliakov, H. Bustince, D. Paternain, Image reduction using means on discrete product lattices, IEEETrans. ImageProcess. 21 (2012) 1070-1083.

[13] X. Liang, W. Xu, Aggregation method for motor drive systems, Eletric Power System Research 117 (2014) 27-35.

[14] D. Dubois, H. Prade (Eds.), Fundamental sof Fuzzy Sets, Kluwer Academic Publishers, Dordrecht, 2000.

[15] G. Beliakov, A. Pradera, T. Calvo, Aggregation functions: a guide for practitioners, Stud. Fuzziness Soft Comput. 221, 2007.

[16] M. Grabisch, E. Pap, J. L. Marichal, R. Mesiar, Aggregation Functions, University Press Cambridge, 2009.

[17] M. Baczyńnski, B. Jayaram, Fuzzy Implications, Springer, Berlin, 2008.

[18] R. R. Yager, Centered OWA operators, Soft Comput. 11 (2007) 631-639.

[19] A. Jurio, M. Pagola, R. Mesiar, G. Beliakov, H. Bustince, Image magnification using interval information, IEEE Trans. Image Process. 20 (2011) 3112-3123.

[20] J. Yang, J. Wright, T. S. Huang, Y. Ma, Image Super-Revolution Via Sparse Representation, IEEE Trans on Image Processing, v. 19, n. 11 (2010) 2861-2873.

[21] J. Yang, J. Wright, T. S. Huang, Y. Ma, Image Super-Revolution as Sparse Representation of Raw Image Patches, Computer Vision and Pattern Recoginition, 2008. CVPR 2008. IEEE Conference, IEEE, 2008, 1-8.